Volume 4, Issue 1, January 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Software Reliability Growth Model

Gaurav Aggarwal
Research Scholar & Jagannath University
India

Dr. V.K Gupta
Assistant Professor & University of Ragasthan, Jaipur
India

Abstract: Software Reliability Model is categorized into two, one is static model and the other one is dynamic model. Dynamic models observe the temporary behavior of debugging process during testing phase. In Static Models, modeling and analysis of program logic is done on the same code. A Model which describes about error detection in software Reliability is called Software Reliability Growth Model. This paper reviews various existing software reliability models and there failure intensity function and the mean value function. On the basis of this review a model is proposed for the software reliability having different mean value function and failure intensity function.

Keywords: Software reliability, Logistic Growth, Curve Model, Software Reliability Model, Mean Value Function, Failure Intensity Function.

I. INTRODUCTION

Software reliability is about to define the stability or the life of software system with different properties. These properties include the trustfulness of software system, software cost, execution time, software stability etc. The aspects related to these software system includes the probability of software faults, frequency of fault occurrence, criticality of fault, associated module with respective fault etc. In a software development process, the pre estimation of software reliability is required to deliver the software product. According to the required level of software quality estimation of software cost, development time is also estimated. There are number of quality measure that approves the software reliability **[1]**. Each stage of software life cycle itself takes some time quantum to deal with software reliability. Higher the software quality, lesser the software maintainability.

Software reliability growth models, refers to those models that try to predict software reliability from test data **[2]**. These models try to show a relationship between fault detection data (i.e. test data) and known mathematical functions such as logarithmic or exponential functions. The goodness of fit of these models depends on the degree of correlation between the test data and the mathematical function **[3]**.

The software reliability is defined as the probability that the software will operate without a failure under a given environmental condition during a specified period of time **[4]**. The software reliability assessment is one of the most important processes during the software development. Since 1970, many software reliability growth models (SRGMs) have been proposed.

In general, there are two major types of software reliability models: the deterministic and the probabilistic. The deterministic one is employed to study the number of distinct operators and operands in the program. The probabilistic one represents the failure occurrences and the fault removals as probabilistic events **[5]**.

The probabilistic models can be further classified into different classes, such as error seeding, failure rate, and non-homogeneous Poisson process (NHPP). Among these classes, the NHPP models are the most popular ones. The reason is the NHPP model has ability to describe the software failure phenomenon. The first NHPP model, which strongly influences the development of many other models presented a NHPP model with S-shaped mean value function. They **[6, 7]** also made further progress in various S-Shaped NHPP models. Although these NHPP models are widely used, they impose certain restrictions or a priori assumptions about the nature of software faults and the stochastic behavior of software failure process.

II. NEURAL NETWORK

Neural networks are a computational metaphor inspired studies of the brain and nervous system in biological organisms. They are highly idealized mathematical models of how understand the essence of these simple nervous systems. The basic characteristics of a neural network **[8]** are

- It consists of many simple processing units, called neurons that perform a local computation on their input to produce an output.
- Many weighted neuron interconnections encode the knowledge of the network.
- The network has a learning algorithm that lets it automatically develop internal representations

One of the most widely used processing unit models is based on the logistic function. The resulting transfer function is given by

$$output = \frac{1}{1 + e^{-sum}}$$

Where sum is the aggregate of weighted inputs. Figure 1 shows the actual I/O response of this unit model. The unit is nonlinear and continuous. There exists a variety of neural network models and learning procedures. Two well-known classes of neural networks that can be used for prediction applications are: feed-forward networks and recurrent networks. They use feed-forward networks and learning procedure is back propagation algorithm which comes under the category of supervised learning.

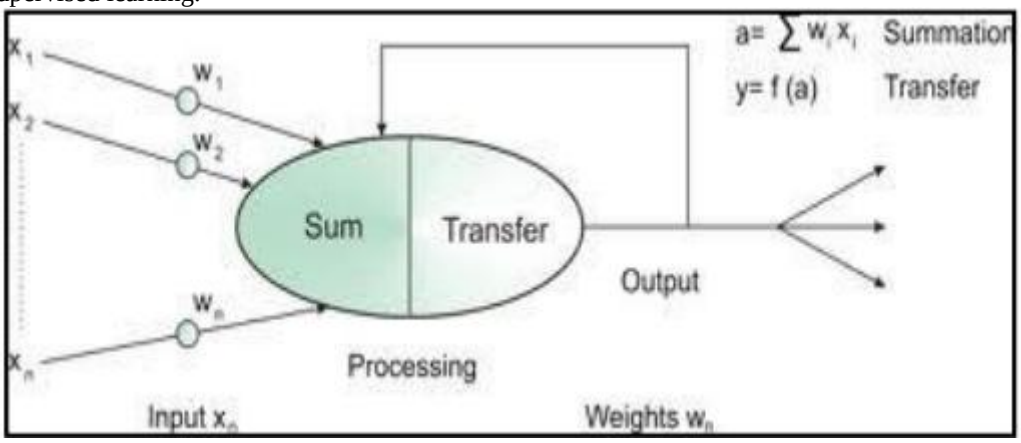

Fig. 1 A typical transfer unit [8]

III. SOFTWARE RELIABILITY GROWTH MODELS (SRGMS) AND CRITERIA

A software reliability growth model (abbreviated as SRGM) is known as one of the fundamental technologies for quantitative software reliability assessment, and playing an important role in software project management for producing a highly-reliable software system**[9]**. SRGM is mathematical model, shows how software reliability improves as faults are detected and repaired. SRGM can be used to predict when a particular level of reliability is likely to be attained. Thus, SRGM is used to determine when to stop testing to attain a given reliability level **[10]**. There are many software reliability growth models but the commonly used model of software reliability models are JM, GO model, MO model, Sch model, S-Shape model. To evaluate the prediction powers of different models, it is necessary to use a meaningful measures. They use two criteria: Root Mean Square Error (RMSE) and Average Error(AE). These criteria are used to measure the difference between the actual and predicted values, the formulas is as Eq. (1) (2).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(c(k) - \hat{c}(k))^2} \quad (1)$$

$$AE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{c(k)-\hat{c}(k)}{c(k)}\right| \times 100 \quad (2)$$

Where n is the number of groups of failure data, $c(k)$ is the number of the actual failures in each group of failure data, $\hat{c}(k)$ is the number of the predicted failures. The smaller the RMSE and AE, the stronger that the model prediction ability **[2]**.

A. Various types of Models

1) ***Jelinski-Moranda Model:*** The Jelinski-Moranda model was first introduced as a software reliability growth model in Jelinski and Moranda (1972) **[11]**. This is a continuous time-independently distributed inter failure times and independent and identical error behavior model **[12]**. The software failure rate of hazard function at any time is proportional to the current fault content of the program. The distribution of the order statistics is the Exponential distribution **[11]**.

The main assumptions for the Jelinski-Moranda model are the following:

(1) At the beginning of testing, there are n_0 faults in the software code with n_0 being an unknown but fixed number.

(2) All faults are of the same type.

(3) Immediate and perfect repair of faults.

(4) Faults are detected independently of each other.

(5) The times between failures are exponentially distributed with parameter proportional to the number of remaining faults.

(6) Each fault is equally dangerous with respect to the probability of its instantaneously causing a failure. Furthermore, the hazard rate of each fault does not change over time, but remains constant at φ.

(7) The failures are not correlated, i.e. given n_0 and φ the times between failures $(\Delta t_1, \Delta t_2, \ldots, \Delta t n_0)$

(8) Whenever a failure has occurred, the fault that caused it is removed instantaneously and without introducing any new fault into the software.

$$z(\Delta t|t_{i-1}) = \phi[n_0 - M(t_{i-1})] = \phi[n_0 - (i-1)] \quad (1)$$

The failure intensity function is the product of the inherent number of faults and the probability density of the time until activation of a single fault, $n_a(t)$, i.e.:

$$\frac{dm(t)}{d(t)} = n_0[1 - \exp(-\phi t)] \quad (2)$$

Therefore, the mean value function is

$$m(t) = n_0[1 - \exp(-\phi t)] \quad (3)$$

It can easily be seen from equations (2) and (3) that the failure intensity can also be expressed as

$$\frac{dm(t)}{d(t)} = \phi[n_0 - m(t)] \quad (4)$$

According to equation (4), the failure intensity of the software at time t is proportional to the expected number of faults remaining in the software; again, the hazard rate of n individual faults is the constant of proportionality. Moreover, many software reliability growth models can be expressed in a form corresponding to equation (4).

One of the most widely discussed assumptions of the Jelinski-Moranda model is (2) since it implies that each repaired fault reduces the hazard rate of the new time between failure by a constant $\lambda > 0$. This idea is depicted in Figure 2.

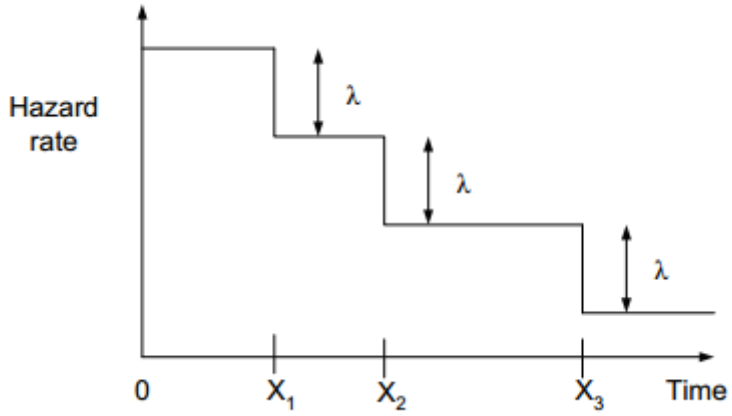

Fig. 2 Jelinski- Moranda model hazard rate. It remains constant between failure observations and decreases by a factor λ after a fault is repaired [11]

*2) **Goel-Okumoto Model***

This model, first proposed by Goel and Okumoto, is one of the most popular NHPP model in the field of software reliability modeling. It is also called the exponential NHPP model. Assumptions (2), (3) and (4) for the Jelinski-Moranda model are also valid for the Goel-Okumoto model. Considering failure detection as a Non homogeneous Poisson process with an exponentially decaying rate function, the mean value function is hypothesized in this model as **[13]**

$$m(t) = a(1 - \exp[-bt]), \qquad a > 0, b > 0$$

and the intensity function of this model is given as

$$\lambda(t) = ab * \exp[-bt], \qquad a > 0, b > 0$$

where a is the expected total number of faults to be eventually detected and b represents the fault detection rate.

In fact, it follows that **[11]**

$$\lim_{t \to \infty} m(t) = a$$

A typical plot of $m(t)$ for the Goel-Okumoto model can be observed in Figure 3 where $m(t)$ is plotted when $a = 11$ and $b = 0.14$. Note that a determine the scale and b the shape of the mean-value function.

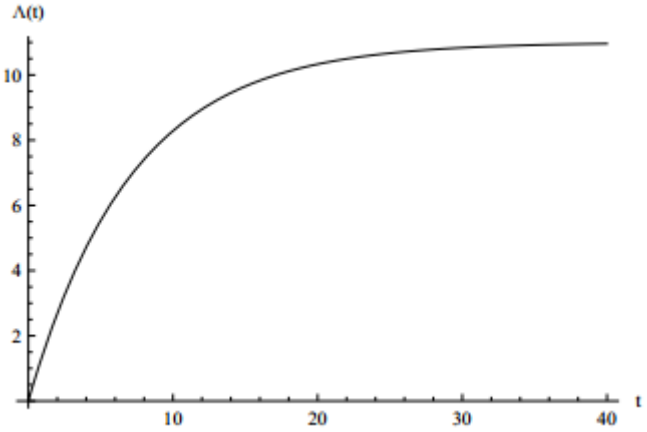

Fig 3: Goel- Okumoto model mean-value function when a = 11 and b = 0.14 [11]

*3) **Generalized Goel NHPP Model***

In order to describe the situation that software failure intensity increases slightly at the beginning and then begins to decrease, Goel proposed a simple generalization of the Goel-Okumoto model with an additional parameter c **[13]** . The mean value function and intensity function are

$$m(t) = a(1 - \exp[-bt^c]), \qquad a > 0, b > 0, c > 0$$
$$\lambda(t) = abct^{c-1} \exp[-bt^c], \qquad a > 0, b > 0, c > 0$$

where a is the expected total number of faults to be eventually detected and b and c are parameters that reflect the quality of testing.

*4) **Inflected S-Shaped Model***

This model solves a technical problem in the Goel-Okumoto model. It was proposed by Ohba and its underlying concept is that the observed software reliability growth becomes S-shaped if faults in a program are mutually dependent, i.e., some faults are not detectable before some others are removed. The mean value function is **[13]**

$$m(t) = a * \frac{1 - \exp[-bt]}{1 + \psi(\mathrm{r}) * \exp[-bt]}, \qquad \psi(r) = \frac{1-r}{r}, \qquad a > 0, b > 0, r > 0$$

The parameter r is the inflection rate that indicates the ratio of the number of detectable faults to the total number of faults in the software, a is the expected total number of faults to be eventually detected, b is the fault detection rate, and is the inflection factor. On taking $\psi(\mathrm{r}) = \beta$ then the inflection S-shaped model mean value function and intensity function are given as

$$m(t) = a * \frac{1 - \exp[-bt]}{1 + \beta * \exp[-bt]}, \qquad a > 0, b > 0, \beta > 0$$

$$\lambda(t) = \frac{ab\exp[-bt](1 + \beta t)}{(1 + \beta * \exp[-bt])^2}, \qquad a > 0, b > 0, \beta > 0$$

*5) **Logistic Growth Curve Model***

In general, software reliability tends to improve and it can be treated as a growth process during the testing phase. That is, the reliability growth occurs due to fixing faults. Therefore, under some conditions, the models developed to predict economic population growth could also be applied to predict software reliability growth. These models simply fit the cumulative number of detected faults at a given time with a function of known form. Logistic growth curve model is one of them and it has an S-shaped curve. Its mean value function and intensity function are **[13]**

$$m(t) = \frac{a}{1 + \mathrm{k} * \exp[-bt]}, \qquad a > 0, b > 0, \mathrm{k} > 0$$

$$\lambda(t) = \frac{ab\exp[-bt]}{(1 + \mathrm{k} * \exp[-bt])^2}, \qquad a > 0, b > 0, \mathrm{k} > 0$$

where a is the expected total number of faults to be eventually detected and k and b are parameters which can be estimated by fitting the failure data.

6) Musa-Okumoto Model

Musa-Okumoto have been observed that the reduction in failure rate resulting from repair action following early failures are often greater because they tend to the most frequently occurring once, and this property has been incorporated in the model **[13]**. The mean value function and intensity function of the model given as

$$m(t) = a * \ln(1 + bt), \qquad a > 0, b > 0$$

$$\lambda(t) = \frac{ab}{(1 + bt)}, \qquad a > 0, b > 0$$

where a is the expected total number of faults to be eventually detected and b is the fault detection rate.

TABLE I: MEAN VALUE AND INTENSITY OF VARIOUS MODELS

Model Name	Mean Value Function	Intensity Function
1. Jelinski-Moranda model	$m(t) = n_0[1 - \exp(-\phi t)]$	$\lambda i = (N - k)\mu$
2. Goel-Okumoto Model	$m(t) = a(1 - exp[-bt]), a > 0, b > 0$	$\lambda(t) = ab * exp[-bt], a > 0, b > 0$
3. Generalized Goel NHPP Model	$m(t) = a(1 - exp[-bt^c]),$ $a > 0, b > 0, c > 0$	$\lambda(t) = abct^{c-1}\, exp[-bt^c],$ $a > 0, b > 0, c > 0$
4. Inflected S-Shaped Model	$m(t) = a * \frac{1 - exp[-bt]}{1 + \psi(r) * exp[-bt]},$ $\psi(r) = \frac{1-r}{r}, \quad a > 0, b > 0, r > 0$	$\lambda(t) = \frac{abexp[-bt](1 + \beta t)}{(1 + \beta * exp[-bt])^2},$ $a > 0, b > 0, \beta > 0$
5. Logistic Growth Curve Model	$m(t) = \frac{a}{1 + k * exp[-bt]},$ $a > 0, b > 0, k > 0$	$\lambda(t) = \frac{abexp[-bt]}{(1 + k * exp[-bt])^2},$ $a > 0, b > 0, k > 0$
6.Musa-Okumoto Model	$m(t) = a * ln(1 + bt), \quad a > 0, b > 0$	$\lambda(t) = \frac{ab}{(1 + bt)}, \quad a > 0, b > 0$

IV. PROPOSED MODEL

On the basis of the model discussed the logistic growth curve model seems perform better than the other models. The model can be designed on the basis of the tangential function. The tangential model must be drawn in the positive axis. It

varies from 0 to infinity similar to the software reliability. The software reliability is inversely proportional to the fault detection as the no of fault detection decrease the reliability increases. The zero fault detection means the infinite reliability and the zero software reliability means the infinite faults. The proposed model suits the behavior of the software reliability so fits to the software reliability.

In the beginning of testing, there is exponential number of faults in the software code. The number of faults is unknown but they are fixed in number. All faults are of same type. Each fault can be detected independent of each other. The remaining number of fault and the remaining time is useful to determine the other parameters. The probability of occurring of each fault is same. Each fault occurred can be removed instantaneously. The mean value function can be given as

$$m(t) = \left(fv = \left(\frac{[1-\exp(-\phi t)]}{[1+\exp(-\phi t)]} \right) \right) > 0 ? fv : 0 \quad (3)$$

The failure intensity can be expressed as

$$\lambda(t) = \frac{dm(t)}{d(t)} \quad (4)$$

According to the failure intensity of the software at time t is proportional to the expected number of faults remaining in the software.

V. CONCLUSION

When a software system is designed, the major concern is the software quality. The quality of software depends on different factors such as software reliability, efficiency, cost etc. This paper study various existing software reliability model with there failure intensity function and the mean value function. This paper also proposes a new model for software reliability having different failure intensity function and mean value function. In future the proposed can be implemented and results of the model can be compared with the existing model results.

REFERENCES

[1] Garima Chawla et. al. , *A Fault Analysis based Model for Software Reliability Estimation*, International Journal of Recent Technology and Engineering (IJRTE),ISSN: 2277 -3878, Volume-2, Issue-3, July 2013.

[2] Rita G. Al gargoor et. al. , *Software Reliability Prediction Using Artificial Techniques*, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 2, July 2013,ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784,www.IJCSI.org.

[3] Al-Rahamneh Z., Reyalat M. Sheta A. F., Bani-Ahmad S., and Al-Oqeili S., *A New Software Reliability Growth Model: Genetic-Programming-Based Approach* , Journal of Software Engineering and Applications, 2011, pp. 476-481.

[4] M. R. Lyu, *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.

[5] Su, Yu Shen, et al. *An Artificial Neural-Network-Based Approach To Software Reliability Assessment*, TENCON 2005, IEEE Region 10. IEEE, 2005

[6] S. Yamada, and S. Osaki, *Reliability Growth Models for Hardware and Software Systems Based on Non-homogeneous Poisson Processes: a Survey*, Microelectronics and Reliability, 23, 1983, pp. 91-112.

[7] S. Yamada, and S. Osaki, *S-shaped Software Reliability Growth Model with Four Types of Software Error Data*, Int. J. Systems Science, 14, 1983, pp. 683-692.

[8] Mamta Arora et. al. , *Software Reliability Prediction Using Neural Network*, International Journal of Software and Web Sciences 5(2), June-August, 2013, pp.88-92.

[9] Inoue S., and Yamada S ,*Two-Dimensional Software Reliability Measurement Technologies*, IEEE , 2009, pp. 223 – 227.

[10] Quadri S. M., Ahmad N. and Farooq S. U. ,*Software Reliability Growth Modeling With Generalized Exponential Testing –Effort And Optimal Software Release Policy* , Global Journal of Computer Science and Technology , 2011, pp.27 – 42.

[11] Prof. C. J. van Duijn et. al. ,*Statistical Procedures for Certification of Software Systems*, © Corro Ramos, Isaac 2009.

[12] Latha Shanmugam et. al., *A Comparison Of Parameter Best Estimation Method For Software Reliability Models*, International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.5, September 2012.

[13] Mohd. Anjum et. al. , *Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value*, I.J. Information Technology and Computer Science, 2013, 02, 1-14.

